IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Application No.	:	10/727,264
Applicants	:	Bret Alan Gorsline et al.
Filed	:	December 3, 2003
Title	:	Methods and Systems for Programmably Generating Electronic Aggregate Creatives for Display on an Electronic Network
TC/A.U.	:	2178
Examiner	:	David Faber
Docket No.	:	002566-73 (019000)
Customer No.	:	64313

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

37 C.F.R. § 1.131 DECLARATION OF RON HYMAN ROTHMAN

Dear Sir:

I, Ron Hyman Rothman, a joint inventor of pending U.S. Patent Application No. 10/727,264, entitled "Methods and Systems for Programmably Generating Electronic Aggregate Creatives for Display on an Electronic Network," hereby declare:

1. I am a joint inventor named in United States Patent Application Serial No. 10/727,264 ("the '264 application").

2. The '264 application generally pertains to the programmable generation of electronic creatives for display to users of an electronic network. I co-invented the subject matter of the claims of the '264 application prior to July 28, 2003, which is the filing date of U.S. Provisional Patent Application No. 60/490,741 (Yasnovsky et al. '741), to which U.S. Patent Application No. 10/700,837 (Yasnovsky et al. '837), filed November 3, 2003, claims priority.

3. As detailed below, the subject matter of the '264 application was conceived prior to July 28, 2003, the filing date of Yasnovsky et al. '741.

4. Exhibit A is a true printout of an invention description that was prepared for internal use within CNET Networks, Inc., prior to July 28, 2003. Exhibit A provides descriptions and figures for embodiments of the claimed invention. For example, Figure 2 shows a diagram for automatically generating aggregate creatives corresponding to the concepts described in the present application with reference to FIGS. 4-7.

5. In view of Exhibit A, the subject matter of the '264 application was conceived prior to July 28, 2003, the filing date of Yasnovsky et al. '741. Additionally, from prior to July 28, 2003 to at least until December 3, 2003, the filing date of the '264 application, I also continued to diligently investigate and refine as a joint inventor the technology that is the subject matter of the '264 application.

6. In addition to work on the technology from prior to July 28, 2003 to December 3, 2003, I continued to diligently work with Patent Counsel on developing as a joint inventor the patent application that resulted in the '264 application.

7. I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and, further, that these statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the '264 application or any patent issued thereon.

Dated: 8/22/08

Ron Hyman Rothman

Ron Hyman Rothman

EXHIBIT A

Automatically Generated Aggregate Creatives

Bret Gorsline, Ron Rothman
bretg@cnet.com, ronr@cnet.com

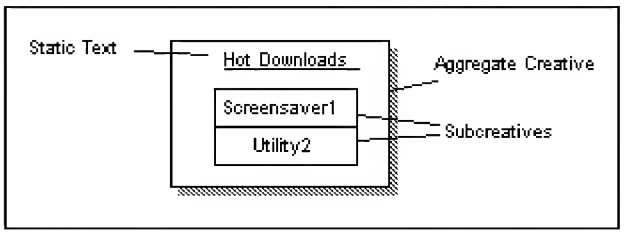
1. Aggregate Advertising Creatives

Aggregate advertising creatives are common on the internet: at a basic level they're just blocks of HTML that contain several paid advertisements. They can appear pretty much anywhere: articles, doors, search results pages, etc. And they could in theory be implemented in several ways: application servers, web server includes, ad systems, etc.

This paper focuses on uses of aggregate creatives that require the features offered by an *ad system*: scheduling, separate tracking, robot filtering, and weighted rotation.

The basic elements of an aggregate creative are shown in Figure 1. It shows an example creative that highlights two downloadable programs. The aggregate creative is comprised of both *static text* and one or more *subcreatives*.

Figure 1. Components of a Sample Aggregate Creative

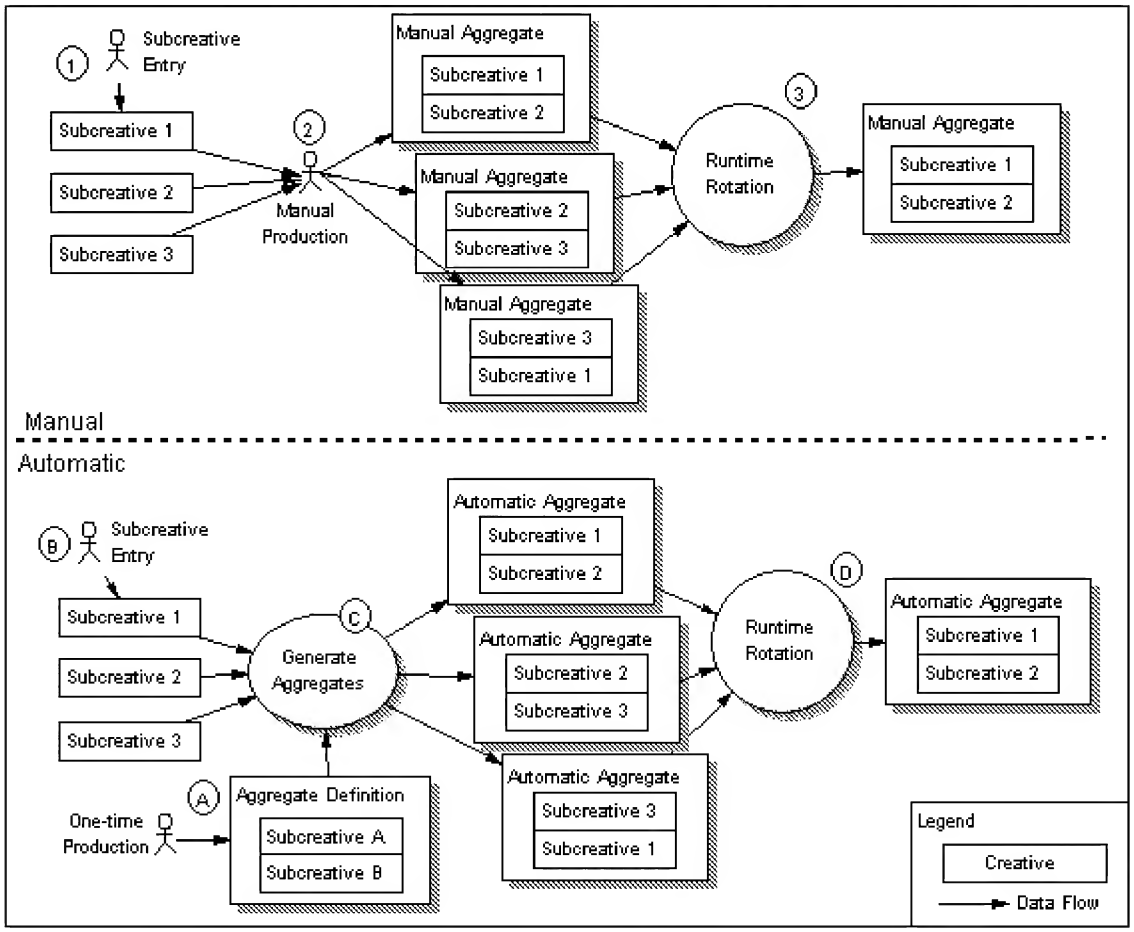


2. Automatic Aggregate Creative - Summary

An *Automatic* Aggregate Creative is a block of ad content constructed from pools of subcreatives scheduled through the ad system. Put another way: they're ads which "pull in" a set of other ads.

Perhaps the best way to introduce automatic aggregates is to contrast them with the static variety. Figure 2 shows the basic flows involved in two cases.

Figure 2. Manual vs Automatic Aggregate Creatives



The upper half of the diagram shows an implementation of **manual** aggregates.[1] The numbered circle (1) shows an ad administrator providing the details of the subcreatives (the creative text, creative schedule, etc.) to another administrator (2) who manually enters several different creatives into the ad system, each one containing a permutation of the subcreatives. The tracking element strings for both impressions and clicks, which are critical—and difficult to get right manually—are also entered. At runtime, the ad delivery system (3) chooses between the forms. Anytime a change is needed to any of the subcreatives, significant manual effort is required to make the appropriate updates in the all of the corresponding aggregates.

The diagram's lower half illustrates the key differences in the **automatic** approach: (A) a definition for the aggregate is produced *once*. From that point on, the only manual input is (B) the entry of the subcreatives and their schedules—again, *once* (per subcreative). The system automatically generates (C) the appropriate permutations for (D) display at

runtime. If any change is required to any of the subcreatives, the subcreative need be updated only once; subsequent automatic regeneration of the aggregates will pick up the updated subcreative.

The manual approach is workable on a small scale, but in practice, manual creation of many permutations or frequent updates are time-consuming and error prone. It is the desire to scale up to hundreds of subcreatives that led us to develop automatic aggregates.

In order to achieve operational scalability, it was determined the system had to have several properties:

1. The initial definition of a dynamic aggregate creative must be reusable for long periods and in different content areas.
2. Subcreative entry must work the same as the entry of any other, "normal" creative
3. Allow the same subcreatives to scheduled at any ad tier, e.g., channel-level, subchannel-level, etc.

The remainder of this paper details the solution implemented and launched as part of Madison 1.1.

Review of key points:

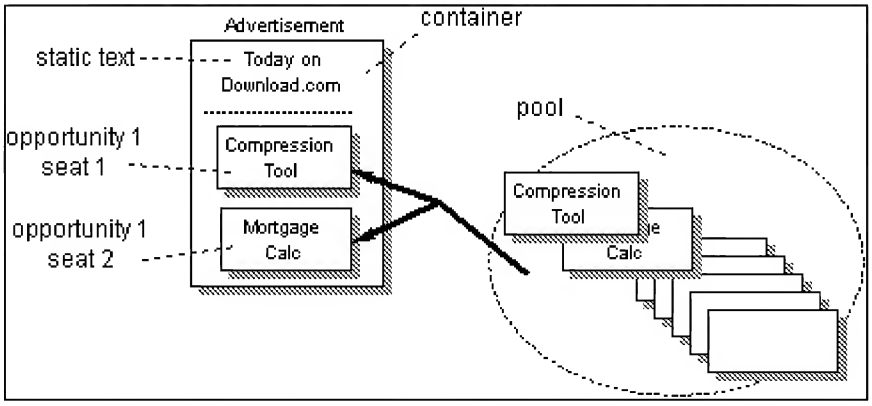
- A key innovation of Automatic Aggregate Creatives is the frequent offline regeneration of the aggregate forms.
- The key benefit is the significant reduction of human time in the production and maintenance of aggregate creatives.

3. Aggregate Creatives In Depth

From here forward, the term "aggregate creative" will be shorthand for "automatic aggregate creative." Some examples and some discussion of terminology will be helpful before delving into some of the technical challenges surrounding aggregate creatives.

Figure 3 shows an idealized view of an important upcoming application of aggregate creatives – the *Download Launch Pad*. It's a block of HTML on Download.com doors, highlighting recent software listed on the site. When manual aggregates were used to implement it, the space was not fully utilized. The sales vision was to rotate a much larger number of software highlights through the box than was feasible to do manually.

Figure 3. Aggregate Creative Example 1: Download Launch Pad



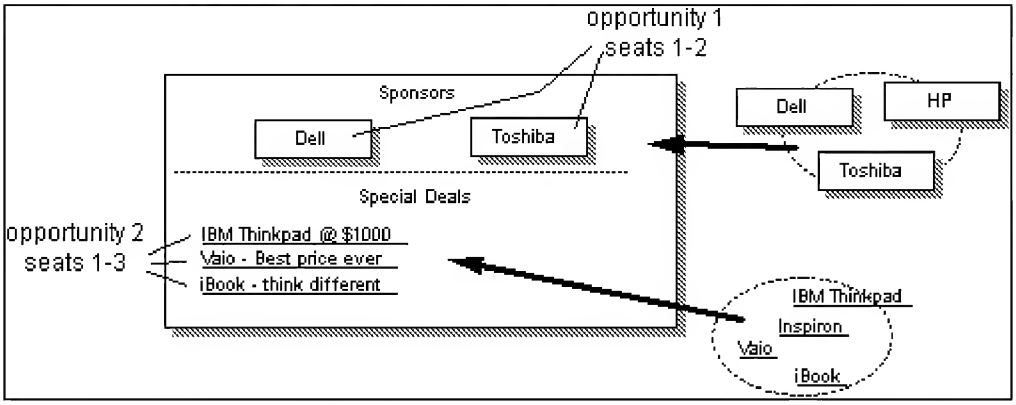
In this example, a definition is created for a single-*opportunity* aggregate with two *seats*. It draws two elements at a time from a *pool* of subcreatives, which is in this case a large set of buttons. (See the glossary for more precise definitions of italicized words.)

Periodically, the system generates many *forms* of the aggregate. The ad system rotates through these forms at run-time, choosing one of them in response to each suitable ad request. At any given time, every subcreative exists in at least one of the forms, sharing the screen real estate as people hit the pages over time.

The period of aggregate regeneration can range from minutes to hours. Madison 1.1 currently regenerates once per hour as well as on-demand.

A more complicated example is *Sponsored Deals* (see Figure 4). This type of creative contains two separate saleable opportunities. The first is a row of two buttons, and the second a list of three text links. There are two corresponding pools of creatives to fill these seats.

Figure 4. Aggregate Creative Example 2 – Sponsored Deals



These examples will be used in the following sections to illustrate some of the technical aspects of our solution.

3.1. Rotation

At the end of the day, the sales team expects that delivery numbers for the various advertisers will report evenly. And during the day, it would be best if a single advertiser didn't get special positioning preference unless they've paid for that privilege.

There are many possible rotation algorithms, but our solution has implemented a simple *sliding* approach.[2] Active subcreatives in the pool are arranged into a circular list; we traverse around the loop such that each subcreative gets a chance to be first.

> **Rotation Example 1**
> One opportunity, two seats
> Subcreatives in the Pool: A,B,C,D
> Generated Aggregate Forms: AB, BC, CD, DA

This simple algorithm has the properties we're looking for. Each subcreative appears 50% of the time. And half the time it's first in the list, the other half it's in the second position.

Working with multiple pools differs only slightly: we traverse the largest pool once and the smaller pool(s) more than once. The traversal through each list of subcreatives happens *independently* of the other lists. (As an illustration, note in Rotation Example 2 that Pool 1 rotates exactly the same way it did in Rotation Example 1, when it was the only pool.)

> **Rotation Example 2**
> Opportunity 1 has 2 seats
> Opportunity 2 has 3 seats
> Pool 1 has 3 subcreatives: A,B,C
> Pool 2 has 4 subcreatives: 1,2,3,4
> Generated Aggregate Forms: AB-123, BC-234, CA-341, AB-412

Notice that the button grouping AB appears twice. This is because we need four aggregate forms to properly rotate through Pool 2. But this causes a problem: Subcreatives A and B would deliver more than C.

The solution is to (a) randomly shuffle the order of each pool's subcreatives before traversal, and (b) regenerate the (randomized) forms frequently throughout the day. At a single point in time, this doesn't solve the problem—some subcreatives will still get "short-changed." But throughout the day—as the permutations are repeatedly regenerated—the differences will even out.

Our current implementation thus supports only so-called "percentage-based" opportunities. I.e., advertisers get a specified proportion of the traffic, rather than a

particular number of impressions per day. Our Automatic Aggregate architecture does not preclude using impression-based subopportunities, however, and we expect that upcoming improvements to the rotation algorithm will certainly include support for impression-based opportunities.

Review of key points:

- The rotation algorithm should treat the subcreatives fairly with respect to ordering within each opportunity.
- Multiple pools are handled by looping once through the largest pool, and more than once around the smaller pools.
- The rotation algorithm achieves even delivery over the day by shuffling the pool before each generation, and by regenerating frequently throughout the day

3.2. Weighting and Constraints

Weighting

In the previous section we focused on obtaining even delivery across the day for each item in the pool. But what if one subcreative is *supposed* to get more visibility? This is where weights come into play. Since an aggregate's subcreatives are treated the same as "regular" creatives, they may be assigned weights to give them a relative boost. In the context of aggregates, the weight is used to copy the subcreative into the pool list multiple times.

> **Rotation Example 3 - weighting**
> One opportunity, two seats
> Subcreatives in the pool: A,B,C,D
> Subcreative A has a weight of 2
> Randomized pool list: D,A,B,A,C
> Generated Aggregates Forms: DA,AB,BA,AC,CD

Constraints

What if the randomization in Rotation Example 3 turned out differently? Using the same setup, but with a different randomized list, we might get:

> **Rotation Example 4 – weighting part 2**
> Randomized pool list: B,A,A,D,C
> Generated Aggregate Forms: BA,AD,AD,DC,CB

You may notice with surprise that only 4 unique forms are generated—form AA was dropped and form AD was included twice. Intuitively, dropping AA makes sense; we probably don't want to display an aggregate in which the same subcreative appears twice. Dropping it makes sense from a sales point of view, and is formalized in the concept of *constraints*. A constraint is a business rule that may be applied to filter aggregate results. There are many imaginable constraints; the ones we've implemented in Madison 1.1 are:

1. Not-Same-Subcreative: set by default, this constraint disallows the same subcreative from appearing twice in the same aggregate form. This constraint can be turned off if desired.
2. Not-Same-Line: disallows a subcreative from appearing in a form if another subcreative from the same line has already been selected.
3. Not-Same-Advertiser: optionally controls the appearance of multiple subcreatives from the same advertiser.

Of course, any number of constraints may be active at the same time. In Rotation Example 4, the permutation AA runs afoul of the default Not-Same-Subcreative rule and is thus disallowed, causing AD to appear twice.

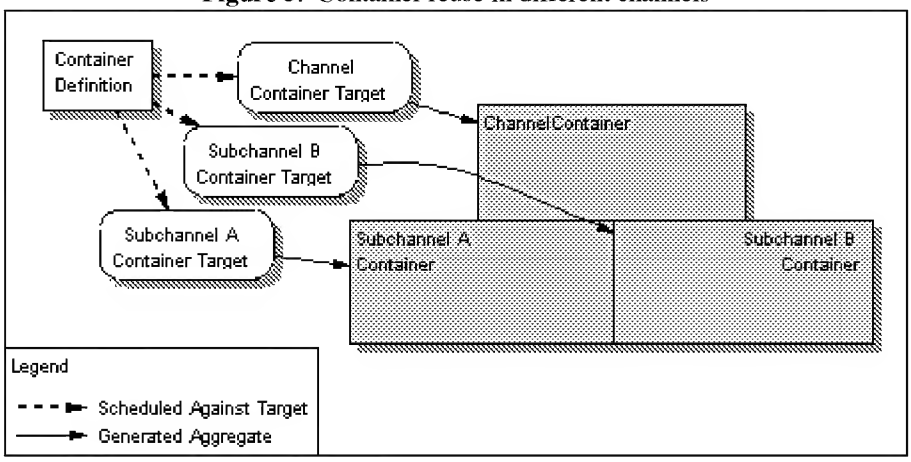
Review of key points:

- Weighting allows administrators to configure particular subcreatives to appear more frequently.
- Constraints are business rules that are be applied to aggregate forms.

3.3. Targeting

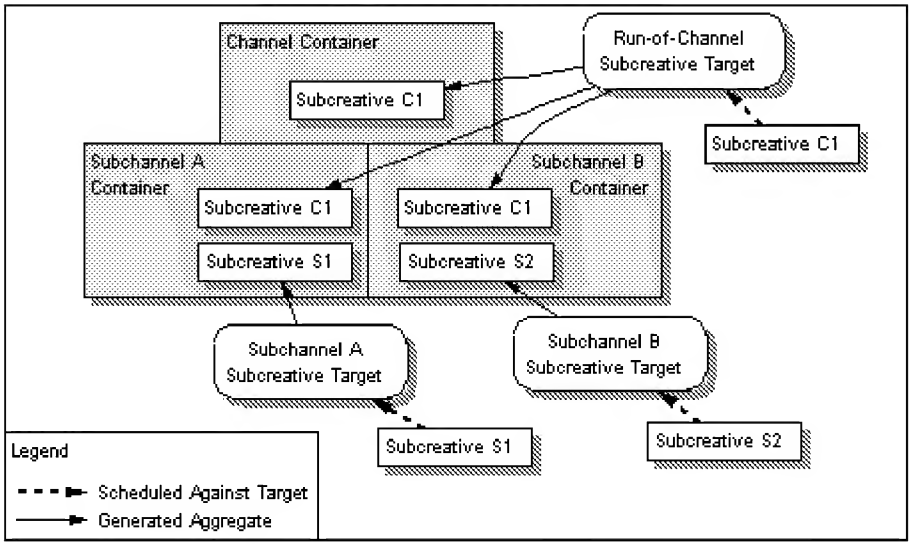
To attain the goal of significantly reducing manual operations, an aggregate creative's container needs to be *reusable* in multiple content areas. (These content areas may reside on different ad targeting tiers.) Thus, instead of designing an aggregate for only one particular channel, it can be associated dynamically with several different targets—a powerful and time-saving feature. Figure 5 shows an example content channel hierarchy.

Figure 5. Container reuse in different channels



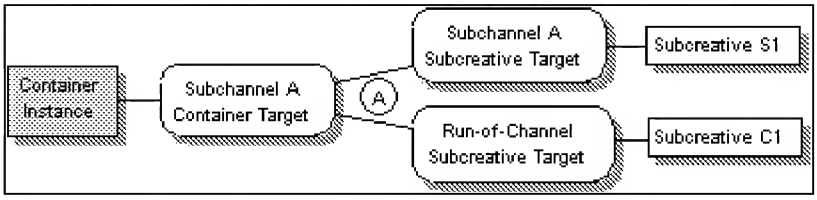
Why do we want to schedule aggregates in this way rather than just running the same container throughout the whole channel? Because it allows more flexibility in targeting the subcreatives: advertisers may want to buy subcreatives which flow either throughout the entire channel or only within select subchannels. Figure 6 illustrates this: advertisers have the choice to buy a subcreative that remains in one particular area (S1 and S2), or one that runs through a larger collection of areas (C1).

Figure 6. Channel-level subcreatives



Supporting the ability to relocate requires a special association in the ad system. In order to generate the correct aggregate forms, the system needs to determine which subcreatives belong in the pools for a particular instance of a container. One way to implement this, as is done in Madison 1.1, is for the ad scheduling system to have a way of associating container targets with subcreative targets. Figure 7 highlights this association at point A.

Figure 7. Finding which subcreatives belong to a container's pool



Determining the population of the pool for a particular aggregate, then, starts by using the container's targeting info to find the subcreative targets that apply. Then the active subcreatives for those targets are gathered into the pool.

Review of key points:

- Containers can be relocated to different content areas, or in hierarchies within the same area.
- Subcreatives can run in one container, or across several containers.
- The ad system must support an association between the container target and the subcreative target.

4. Formatting Concerns

A professional web site team cares deeply about the presentation of the page to the end-user. Truncated or poorly presented HTML is unprofessional and to be avoided.

So what if a pool is empty or doesn't have enough subcreatives to fill the seats in the opportunity? We don't want to end up with an empty box or misaligned seats within an aggregate creative.

The container definition language should have enough power to allow for handling of these special cases, a property we call "graceful degradation." Two special cases are *empty pools* and *short pools*.

If a pool contains no active subcreatives, it is "empty" and the author will likely want to drop the opportunity from the aggregate entirely, or cancel the whole aggregate.

A "short" pool is one where there aren't enough subcreatives to fill all available seats. In these cases, some flexibility in alignment may be important. For instance, when a row of three seats finds only 1 subcreative, the author should be able to specify whether that seat should be centered in the row or put on either side.

Review of key point:

- The container definition language should support graceful degradation of aggregates in the cases of empty or short pools.

5. Conclusion

This paper has presented CNET's innovation of Automatic Aggregate Creatives. The release of this technology with Madison 1.1 will allow the company to realize both cost savings and additional revenue opportunities. Cost savings come from reducing the amount of time the Rich Media team spends crafting the existing manual aggregates. New sales opportunities are possible with the definition of valuable aggregates like the Download Launch Pad, which wouldn't have been scalable before.

Glossary

Aggregate Creative - blocks of HTML that contain several paid advertisements. This paper discusses two main forms: manual and automatic. In line with the main focus, the term Aggregate Creative is used as shorthand for Automatic Aggregate Creative.

Automatic Aggregate Creative – a block of content constructed from pools of subcreatives scheduled through the ad system. Put another way: they're ads which "pull in" a set of other ads. They are comprised of a *container* and one or more *seats* from one or more *opportunities*.

Creative – the visible form of the ad, whether an image or text.

Constraint – business rule that may be applied to filter aggregate results. For instance, "no chosen sub-creatives should have the same advertiser", "… competing advertisers", etc.

Container – the skeletal structure of the aggregate creative. May contain static text, links, etc. Normally wraps around one or more *seats* from paid *opportunities*.

Container Target – the aggregate creative itself is scheduled against an internal administrative account and has no real ratecard entry. e.g. 'Launchpad Box – DL'

Definition – the implementation of the container. May be a template or a program depending on the particular implementation. In Madison 1.1, our definitions are Perl meta-creatives.

Form – The same container segment has many possible display forms. This is a new concept for the ad engine, which takes one more step to determine the form of the segment to deliver to the user.

Graceful Degredation – the property of a container definition language that allows the author of the container to specify special cases like empty or short pools.

Line – a group of creatives from the same advertiser that share a goal.

Opportunity – what the advertiser buys

Opportunity Target – the items sold to advertisers have different targets, but will likely be structured similarly to the Container target. e.g. 'Launchpad link – DL'

Pool – the set of all active creatives scheduled against an opportunity.

Relocation of Containers – the property of containers that allows the same definition to be applied to several different content areas

Rotation – in general, ad rotation is the property that reloading a page causes a different ad to appear. As applied to aggregates, reloading the page may cause different subcreatives of the opportunity to appear in the seats.

Seat – space for a single subcreative from the opportunity

Schedule – as a verb, scheduling is the act of putting dates and goals against a creative. As a noun, it's a combination of the creative and dates.

Short Pool – a pool that doesn't have enough entries to fill all the seats in an opportunity.

Sort – specification of how chosen subcreatives should be arranged in the seats. e.g. by priority, alphabetically, randomly, etc.

Subcreative – ad creatives scheduled inside of aggregates.

Target – a construct in the ad system that defines what content area the ad appears in

Endnotes

[1] This example of manually-produced aggregates is a sophisticated scenario. Given the amount of effort involved, most implementations (including the one at CNET) would likely be much simpler. The optimistic view was taken to highlight just the key benefits—rather than all of the benefits—of automatic aggregates.

[2] We consider it desirable to minimize the number of aggregate forms generated. More sophisticated algorithms that explode the number of permutations to achieve better balance would be possible, but may slow generation and make operational concerns like debugging more difficult.